Transport Area Working Group   J. Saldana
Internet-Draft   University of Zaragoza
Obsoletes: 4170 (if approved)   D. Wing
Intended status: BCP   Cisco Systems
Expires: September 1, 2012   J. Fernandez Navajas
                                                University of Zaragoza
                                                    Muthu A M. Perumal
                                                         Cisco Systems
                                                          G. Camarillo
                                                   Ericsson Research
                                                    February 29, 2012

            Tunneling Compressed Multiplexed Traffic Flows (TCMTF)
                      draft-saldana-tsvwg-tcmtf-01

Abstract

   This document describes a method to improve the bandwidth utilization
   of network paths that carry multiple streams in parallel between two
   endpoints, as in voice trunking. The method combines standard
   protocols that provide compression, multiplexing, and tunneling over
   a network path for the purpose of reducing the bandwidth used when
   multiple streams are carried over that path.

Status of this Memo

   This Internet-Draft is submitted to IETF in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on September 1, 2012.

Table of Contents

1. Introduction

   This document describes a way to combine existing protocols for
   compression, multiplexing, and tunneling to save bandwidth for some
   applications that generate small packets, such as real-time ones.

1.1. Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Bandwidth efficiency of real-time flows

   In the last years we are witnessing the raise of new real-time
   services that use the Internet for the delivery of interactive
   multimedia applications. The most common of these services is VoIP,
   but many others have been developed, and are experiencing a
   significant growth: videoconferencing, telemedicine, video vigilance,
   online gaming, etc.

   The first design of the Internet did not include any mechanism
   capable of guaranteeing an upper bound for delivery delay, taking
   into account that the first deployed services were e-mail, file
   transfer, etc., in which delay is not critical. RTP [RTP] was first
   defined in 1996 in order to permit the delivery of real-time
   contents. Nowadays, although there are a variety of protocols used
   for signaling real-time flows (SIP [SIP], H.323, etc.), RTP has
   become the standard par excellence for the delivery of real-time
   content.

   RTP was designed to work over UDP datagrams. This implies that an
   IPv4 packet carrying real-time information has to include 40 bytes of
   headers: 20 for IPv4 header, 8 for UDP, and 12 for RTP. This
   overhead is significant, taking into account that many real-time
   services send very small payloads. It becomes even more significant
   with IPv6 packets, as the basic IPv6 header is twice the size of the
   IPv4 header (Table 1).

```
+-------------------------------+-------------------------------+
|             IPv4              |             IPv6              |
+-------------------------------+-------------------------------+
|   IPv4+UDP+RTP: 40 bytes header  |  IPv6+UDP+RTP: 60 bytes header  |
|   G.711 at 20 ms packetization:  |  G.711 at 20 ms packetization:  |
|        25% header overhead       |       37.5% header overhead     |
|   G.729 at 20 ms packetization:  |  G.729 at 20 ms packetization:  |
|        200% header overhead      |        300% header overhead     |
+-------------------------------+-------------------------------+
```

Table 1: Efficiency of different voice codecs

In order to mitigate this bad network efficiency, the multiplexing of
a number of payloads into a single packet can be considered as a
solution. If we have only one flow, the number of samples included
in a packet can be increased, but at the cost of adding new
packetization delays. However, if a number of flows share the same
path between an origin and a destination, a multiplexer can build a
bigger packet in which a number of payloads share a common header. A
demultiplexer is necessary at the end of the common path, so as to
rebuild the packets as they were originally sent, making multiplexing
a transparent process for the extremes of the flow.

The headers of the original packets can be compressed to save more
bandwidth, taking into account that there exist some header
compressing standards ([cRTP], [ECRTP], [IPHC], [ROHC]). When
different headers are compressed together, tunneling can be used to
relieve intermediate routers from the decompression and compression
processing.

1.3. Real-time applications not using RTP

But there are many real-time applications that do not use RTP. Some
of them send UDP packets, e.g. First Person Shooter (FPS) online
games, for which latency is very critical. There is also another
fact which has to be taken into account: TCP is getting used for
media delivery. For many reasons, such as avoiding firewalls, the
standard RTP/UDP/IP protocol stack is substituted in many cases by
FLV/HTTP/TCP/IP (FLash Video [FLV]).

There is also another kind of applications which have been reported
as real-time using TCP: MMORPGs (Massively Multiplayer Online Role
Playing Games), which in some cases have millions of players,
thousands of them sharing the same virtual world. They use TCP
packets to send the player commands to the server, and also to send
to the player's application the characteristics and situation of
other gamers' avatars. These games do not have the same
interactivity of FPSs, but the quickness and the movements of the

player are important, and can decide if they win or lose a fight.

1.4. Scenarios of application

Different scenarios of application can be considered for the
tunneling, compressing and multiplexing solution: for example, voice
trunking between gateways of different offices of an enterprise.
Also, the traffic of the users of an application in a town or a
district, which can be multiplexed and sent to the central server.
Also Internet cafes are suitable of having many users of the same
application (e.g. a game) sharing the same access link.

Another interesting scenario are satellite communication links that
often manage the bandwidth by limiting the transmission rate,
measured in packets per second (pps), to and from the satellite.
Applications like VoIP that generate a large number of small packets
can easily fill the limited number of pps slots, limiting the
throughput across such links. As an example, a G.729a voice call
generates 50 pps at 20 ms packetization time. If the satellite
transmission allows 1,500 pps, the number of simultaneous voice calls
is limited to 30. This results in poor utilization of the satellite
link's bandwidth as well as places a low bound on the number of voice
calls that can utilize the link simultaneously. Multiplexing small
packets into one packet for transmission would improve the
efficiency. Satellite links would also find it useful to multiplex
small TCP packets into one packet. This could be especially
interesting for compressing TCP ACKs.

There is still another interesting use case: desktop or application
sharing where the traffic from the server to the client typically
consists of the delta of screen updates. Also, the standard for
remote desktop sharing emerging for WebRTC in the RTCWEB Working
Group is: {something}/SCTP/UDP (Stream Control Transmission Protocol
[SCTP]). In this scenario, SCTP/UDP could be used in other cases:
chatting, file sharing and applications related to WebRTC peers.
There could be hundreds of clients at a site talking to a server
located at a datacenter over a WAN. Compressing, multiplexing and
tunneling this traffic could save WAN bandwidth and potentially
improve latency.

1.5. Objective of this standard

In conclusion, a standard that multiplexes, compresses and sends
packets using a tunnel can be interesting for many enterprises:
developers of VoIP systems can include this option in their
solutions; or game providers, who can achieve bandwidth savings in
their supporting infrastructures. Other fact that has to be taken
into account is that the technique not only saves bandwidth but also

   reduces the number of packets per second, which sometimes can be a
   bottleneck for a satellite link or even for a network router.

   If only one stream is tunneled and compressed, then little bandwidth
   savings will be obtained. In contrast, multiplexing is helpful to
   amortize the overhead of the tunnel header over many payloads.

1.6. Overview of Protocols

   The current standard [TCRTP] defines a way to combine different
   standard protocols. Three layers are considered, as shown in the
   figure:

               RTP/UDP
                  |
                  | ----------------------------
                  |
                ECRTP   compressing layer
                  |
                  | ----------------------------
                  |
                PPPMUX   multiplexing layer
                  |
                  | ----------------------------
                  |
                L2TP   tunneling layer
                  |
                  | ----------------------------
                  |
                 IP
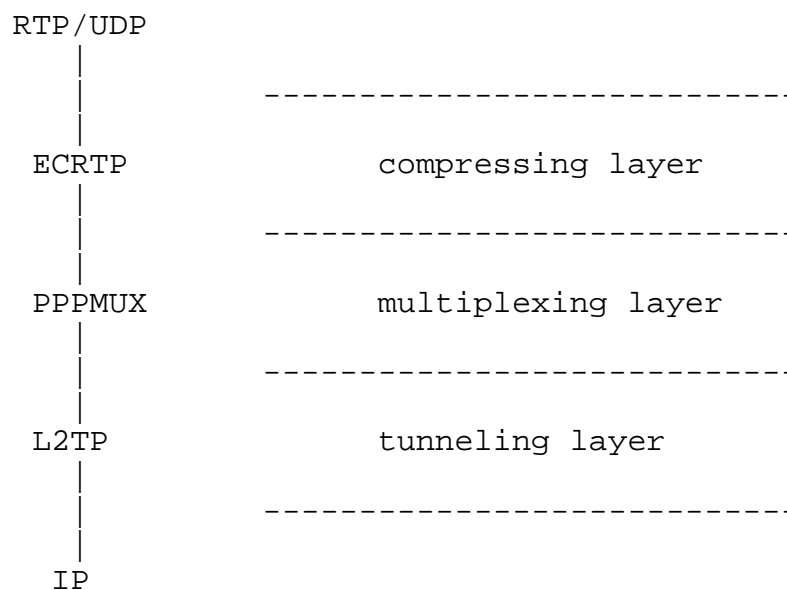

                            Figure 1

   In contrast, the new proposal includes other protocols to be
   compressed in addition to RTP/UDP, since real-time services can also
   be provided using UDP or TCP.

```
       TCP     UDP  RTP/UDP
        |       |      |
         \      |     /                  ----------------------------
          \     |    /
Nothing or ROHC or ECRTP or IPHC                header compressing layer
                |
                |                        ----------------------------
                |
     PPPMUX or other mux protocols              multiplexing layer
                |
                |                        ----------------------------
                |
       GRE or L2TP or other                     tunneling layer
                |
                |                        ----------------------------
               IP
```

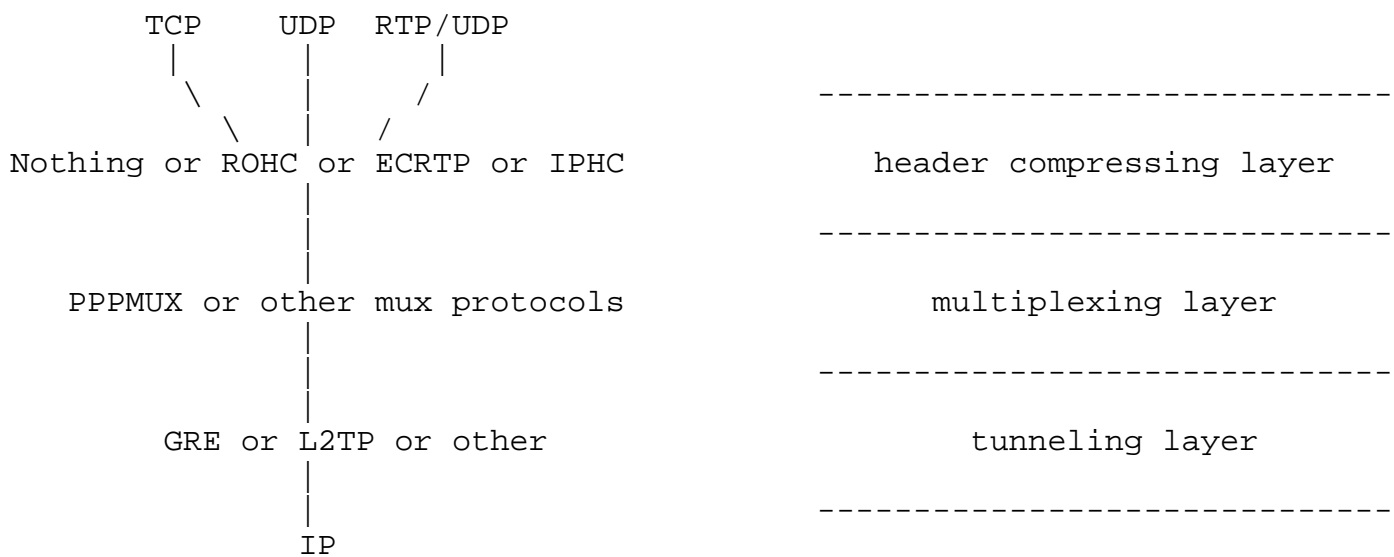   Figure 2

   Each of the three layers is considered as independent of the other
   two, i.e. different combinations of protocols can be implemented
   according to the new standard:

   o Regarding compression, a number of options can be considered: as
      the standards are able to compress different headers, the one to
      be used could be selected depending on the traffic to compress.
      It also exists the possibility of having a null header
      compression, in the case of wanting to avoid traffic compression,
      taking into account the need of storing a context for every flow
      and the problems of context desynchronization in certain
      scenarios. For this, different header compression protocols have
      been defined ([cRTP], [ECRTP], [IPHC], [ROHC]) by the IETF.

   o Multiplexing is accomplished using PPP Multiplexing [PPP-MUX].
      Nevertheless, other multiplexing protocols can also be considered.

   o Tunneling is accomplished by using L2TP (Layer 2 Tunneling
      Protocol [L2TPv3]), GRE (Generic Routing Encapsulation [GRE]) or
      other schemes.

   Payload compression schemes could also be used, but they are not the
   aim of this standard.

2. Protocol Operation

   This section describes how to combine three protocols: compressing,
   multiplexing, and tunneling, to save bandwidth for real-time

applications.

2.1. Models of implementation

TCMTF can be implemented in different ways. The most straightforward
is to implement it in the devices terminating the real-time streams
(these devices can be e.g. voice gateways, or proxies grouping a
number of flows):

```
        [ending device]---[ending device]
                        ^
                        |
                  TCMTF over IP
```
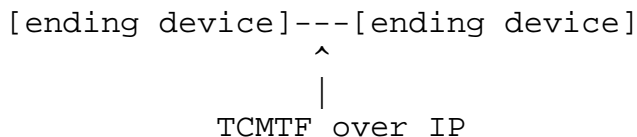
                             Figure 3

Another way TCMTF can be implemented is with an external
concentration device. This device could be placed at strategic
places in the network and could dynamically create and destroy TCMTF
sessions without the participation of the endpoints that generate
real-time flows.

```
   [ending device]\                                  /[ending device]
   [ending device]---[concentrator]---[concentrator]---[ending device]
   [ending device]/                                  \[ending device]
                 ^                  ^                  ^
                 |                  |                  |
            Native IP        TCMTF over IP        Native IP
```
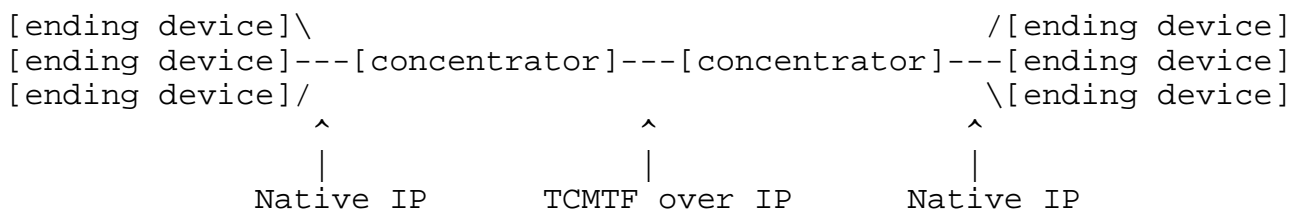
                             Figure 4

Such a design also allows classical compressing protocols to be used
on links with only a few active flows per link.

```
 [ending device]\                                           /[ending device]
 [ending device]---[concentrator]---[concentrator]---[ending device]
 [ending device]/                                           \[ending device]
                 ^                    ^                    ^
                 |                    |                    |
              Compressed         TCMTF over IP        Compressed
```


                                Figure 5
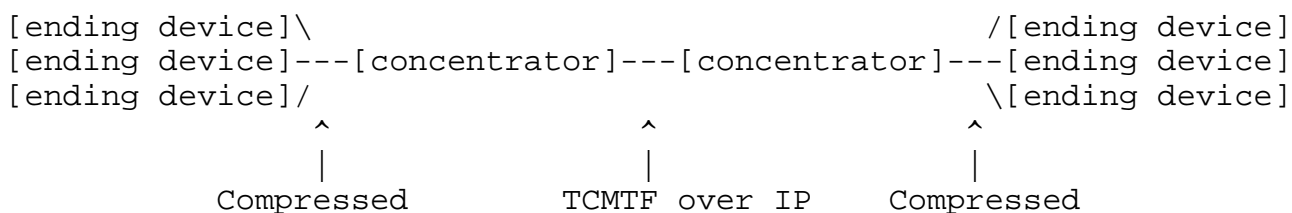

2.2. Choice of the compressing protocol

   There are different protocols that can be used for compressing real-
   time flows:

   o IPHC (IP Header Compression [IPHC]) permits the compression of
      TCP/IP, UDP/IP and ESP/IP headers (Encapsulating Security Payload
      [ESP]). It has a low implementation complexity. On the other
      hand, the resynchronization of the context can be slow over long
      RTT links. It should be used in scenarios presenting very low
      packet loss percentage.

   o cRTP (compressed RTP [cRTP]) works the same way as IPHC, but is
      also able to compress RTP headers. The link layer transport is
      not specified, but typically PPP is used. For cRTP to compress
      headers, it must be implemented on each PPP link. A lot of
      context is required to successfully run cRTP, and memory and
      processing requirements are high, especially if multiple hops must
      implement cRTP to save bandwidth on each of the hops. At higher
      line rates, cRTP's processor consumption becomes prohibitively
      expensive. cRTP is not suitable over long-delay WAN links commonly
      used when tunneling, as proposed by this document. To avoid the
      per-hop expense of cRTP, a simplistic solution is to use cRTP with
      L2TP to achieve end-to-end cRTP. However, cRTP is only suitable
      for links with low delay and low loss. However, once multiple
      router hops are involved, cRTP's expectation of low delay and low
      loss can no longer be met. Further, packets can arrive out of
      order.

   o ECRTP (Enhanced Compressed RTP [ECRTP]) is an extension of cRTP
      [cRTP] that provides tolerance to packet loss and packet
      reordering between compressor and decompressor. Thus, ECRTP
      should be used instead of cRTP.

   o ROHC (RObust Header Compression [ROHC]) is able to compress
      TCP/IP, UDP/IP, ESP/IP and RTP/UDP/IP headers. It is a robust
      scheme developed for header compression over links with high bit
      error rate, such as wireless ones. It incorporates mechanisms for

quick resynchronization of the context. It includes an improved
encoding scheme for compressing the header fields that change
dynamically. Its main drawback is that it requires significantly
more processing and memory resources than the ones necessary for
IPHC or ECRTP.

This standard does not determine which of the existing protocols has
to be used for the compressing layer. The decision will depend on
the scenario, and will mainly be determined by the packet loss
probability, RTT, and the availability of memory and processing
resources. The standard is also suitable to include other
compressing schemes that may be further developed.

2.2.1. Context Synchronization in ECRTP

When the compressor receives an RTP packet that has an unpredicted
change in the RTP header, the compressor should send a COMPRESSED_UDP
packet (described in [ECRTP]) to synchronize the ECRTP decompressor
state. The COMPRESSED_UDP packet updates the RTP context in the
decompressor.

To ensure delivery of updates of context variables, COMPRESSED_UDP
packets should be delivered using the robust operation described in
[ECRTP].

Because the "twice" algorithm described in [ECRTP] relies on UDP
checksums, the IP stack on the RTP transmitter should transmit UDP
checksums. If UDP checksums are not used, the ECRTP compressor
should use the cRTP Headers checksum described in [ECRTP].

2.2.2. Context Synchronization in ROHC

ROHC [ROHC] includes a more complex mechanism in order to maintain
context synchronization. It has different operation modes and
defines compressor states which change depending on link behavior.

2.3. Multiplexing

Header compressing algorithms require a layer two protocol that
allows identifying different protocols. PPP [PPP] is suited for
this, although other multiplexing protocols can also be used for this
layer of TCMTF.

When header compression is used inside of a tunnel, it will reduce
the size of the IP, UDP, and IP headers of the IP packet carried in
the tunnel. However, the tunnel itself has overhead due to its IP
header and the tunnel header (the information necessary to identify
the tunneled payload). One way to reduce the overhead of the IP

header and tunnel header is to multiplex multiple real-time payloads
in a single tunneled packet.

2.3.1. Tunneling Inefficiencies

To get reasonable bandwidth efficiency using multiplexing within an
L2TP tunnel, multiple real-time streams should be active between the
source and destination of an L2TP tunnel. The packet size of the
real-time streams has to be small in order to permit a good bandwidth
saving.

If the source and destination of the L2TP tunnel are the same as the
source and destination of the compressing protocol sessions, then the
source and destination must have multiple active real-time streams to
get any benefit from multiplexing.

Because of this limitation, TCMTF is mostly useful for applications
where many real-time sessions run between a pair of endpoints. The
number of simultaneous sessions required to reduce the header
overhead to the desired level depends on the size of the L2TP header.
A smaller L2TP header will result in fewer simultaneous sessions
being required to produce adequate bandwidth efficiencies.

2.4. Tunneling

L2TP tunnels should be used to tunnel the ECRTP payloads end to end.
L2TP includes methods for tunneling messages used in PPP session
establishment, such as NCP (Network Control Protocol). This allows
[IPCP-HC] to negotiate ECRTP compression/decompression parameters.

Other tunneling schemes, such as GRE [GRE] may also be used to
implement the tunneling layer of TCMTF.

2.5. Encapsulation Formats

The packet format for a packet compressed is:

```
+-----------+----------------------+
|           |                      |
|   Compr   |                      |
|  Header   |         Data         |
|           |                      |
|           |                      |
|           |                      |
+-----------+----------------------+
```

                              Figure 6

The packet format of a multiplexed PPP packet as defined by [PPP-MUX]
is:

```
+-------+---+------+-------+-----+   +---+------+-------+-----+
| Mux   |P L|      |       |     |   |P L|      |       |     |
| PPP   |F X|Len1  |  PPP  |     |   |F X|LenN  |  PPP  |     |
| Prot. |F T|      | Prot. |Info1|  ~|F T|      | Prot. |InfoN|
| Field |   |      | Field1|     |   |   |      |FieldN |     |
| (1)   |1-2 octets| (0-2) |     |   |1-2 octets| (0-2) |     |
+-------+---------+------+-----+   +---------+------+-----+
```

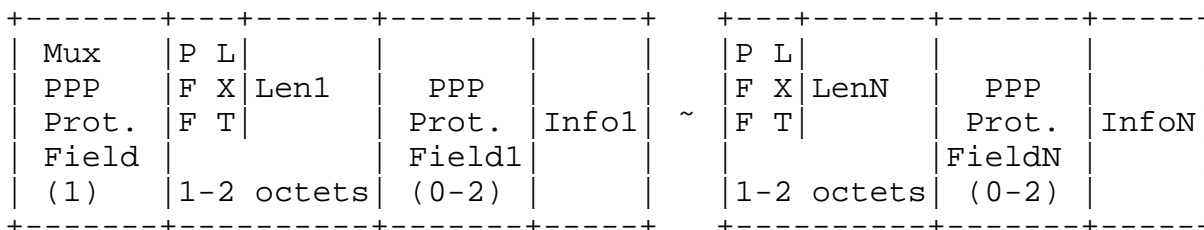   Figure 7

The combined format used for TCMTF with a single payload is all of
the above packets concatenated. Here is an example with one payload:

```
+------+------+-------+----------+-------+--------+----+
| IP   |Tunnel| Mux   |P L|      |       |        |    |
|header|header| PPP   |F X|Len1  |  PPP  | Compr  |    |
| (20) |      | Proto |F T|      | Proto | header |Data|
|      |      | Field |   |      | Field1|        |    |
|      |      | (1)   |1-2 octets| (0-2) |        |    |
+------+------+-------+----------+-------+--------+----+
       |<------------ IP payload -------------------->|
                      |<-------- Mux payload --------->|
```

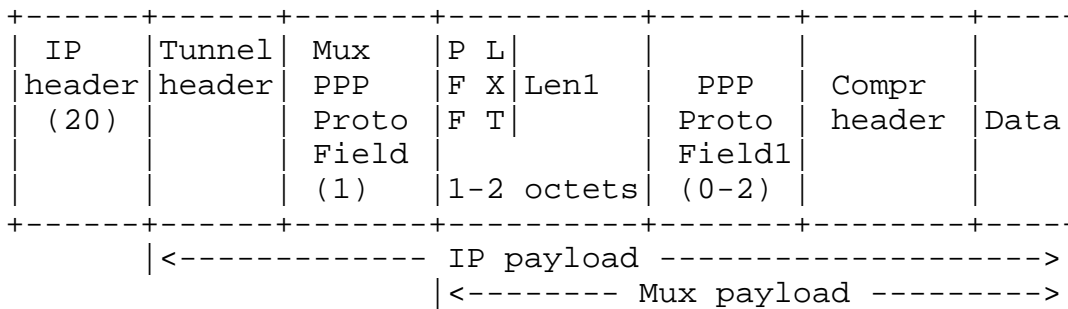   Figure 8

If the tunnel contains multiplexed traffic, multiple "PPPMux
payload"s are transmitted in one IP packet.

3. Contributing Authors

    Jose Ruiz-Mas
    University of Zaragoza
    Dpt. IEC Ada Byron Building
    Zaragoza, 50018
    Spain

    Phone: +34 976762158
    Email: jruiz@unizar.es

    Michael A. Ramalho
    Cisco Systems, Inc.
    1802 Rue de la Porte
    Wall Township, NJ   07719-3784
    US

    Phone: +1.732.449.5762
    Email: mramalho@cisco.com

4. Acknowledgements

5. IANA Considerations

    This memo includes no request to IANA.

6. Security Considerations

    All drafts are required to have a security considerations section.
    See RFC 3552 [RFC3552] for a guide.

7. References

7.1. Normative References

    [ECRTP]   Koren, T., Casner, S., Geevarghese, J., Thompson, B., and
                 P. Ruddy, "Enhanced Compressed RTP (CRTP) for Links with
                 High Delay, Packet Loss and Reordering", RFC 3545, 2003.

    [ESP]   Kent, S., "IP Encapsulating Security Payload", RFC 4303,
                 2005.

   [FLV]   ISO/IEC, "FLV and F4V File Format Specification",
                 14496-12 MPEG-4 Part 12, 2008.

   [GRE]   Farinacci, D., Li, T., Hanks, S., Meyer, D., and P.
                Traina, "Generic Routing Encapsulation (GRE)", RFC 2784,
                2000.

   [I.363.2]   ITU-T, "B-ISDN ATM Adaptation layer specification: Type 2
                AAL", I. 363.2, 1997.

   [IPCP-HC]   Engan, M., Casner, S., Bormann, C., and T. Koren, "IP
                Header Compression over PPP", RFC 3544, 2003.

   [IPHC]   Degermark, M., Nordgren, B., and S. Pink, "IP Header
                Compression", RFC 2580, 1999.

   [L2TPv3]   Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling
                Protocol - Version 3 (L2TPv3)", RFC 3931, 2005.

   [PPP]   Simpson, W., "The Point-to-Point Protocol (PPP)",
                RFC 1661, 1994.

   [PPP-MUX]   Pazhyannur, R., Ali, I., and C. Fox, "PPP Multiplexing",
                RFC 3153, 2001.

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
                Requirement Levels", BCP 14, RFC 2119, March 1997.

   [ROHC]   Jonsson, L-E., Pelletier, G., and K. Sandlund, "The RObust
                Header Compression (ROHC) Framework", RFC 4995, 2007.

   [RTP]   Schulzrinne, H., Casner, S., Frederick, R., and V.
                Jacobson, "RTP: A Transport Protocol for Real-Time
                Applications", RFC 3550, 2003.

   [SCTP]   Stewart, Ed., R., "Stream Control Transmission Protocol",
                RFC 4960, 2007.

   [SIP]   Rosenberg, J., Schulzrinne, H., Camarillo, G., and et.
                al., "SIP: Session Initiation Protocol", RFC 3261, 2005.

   [TCRTP]   Thomson, B., Koren, T., and D. Wing, "Tunneling
                Multiplexed Compressed RTP (TCRTP)", RFC 4170, 2005.

   [cRTP]   Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP
                Headers for Low-Speed Serial Links", RFC 2508, 1999.

7.2. Informative References

   [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC
              Text on Security Considerations", BCP 72, RFC 3552,
              July 2003.

Authors' Addresses

   Jose Saldana
   University of Zaragoza
   Dpt. IEC Ada Byron Building
   Zaragoza,   50018
   Spain

   Phone: +34 976 762 698
   Email: jsaldana@unizar.es

   Dan Wing
   Cisco Systems
   771 Alder Drive
   San Jose, CA 95035
   US

   Phone: +44 7889 488 335
   Email: dwing@cisco.com

   Julian Fernandez Navajas
   University of Zaragoza
   Dpt. IEC Ada Byron Building
   Zaragoza,   50018
   Spain

   Phone: +34 976 761 963
   Email: navajas@unizar.es

Muthu Arul Mozhi Perumal
Cisco Systems
Cessna Business Park
Sarjapur-Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Phone: +91 9449288768
Email: mperumal@cisco.com

Gonzalo Camarillo
Ericsson Research
Jorvas,
Finland

Phone:
Email: gonzalo.camarillo@ericsson.com